

Application Program Interface Supplement  
to the  
Software Communications Architecture Specification

**APPENDIX H**

**I/O Building Block Service Definition**

Revision Summary

1.0	Initial Release
2.2.1	Document numbering change for consistency with SCA main document numbering.

## Table Of Contents

<b>H.1 INTRODUCTION.....</b>	<b>H-1</b>
H.1.1 OVERVIEW .....	H-1
H.1.2 SERVICE LAYER DESCRIPTION.....	H-2
H.1.3 MODES OF SERVICE.....	H-2
H.1.4 SERVICE STATES.....	H-3
H.1.5 REFERENCED DOCUMENTS.....	H-3
<b>H.2 UUID. ....</b>	<b>H-3</b>
<b>H.3 SERVICES.....</b>	<b>H-3</b>
H.3.1 I/O CONFIGURATION BUILDING BLOCK.....	H-3
H.3.1.1 Configuration.....	H-4
H.3.2 I/O CONTROL BUILDING BLOCK.....	H-4
H.3.2.1 parameters.....	H-4
H.3.2.2 txActive.....	H-5
H.3.2.3 enableRTSCTS.....	H-5
H.3.2.4 setCTS.....	H-5
H.3.3 AUDIBLE ALERTS AND ALARMS BUILDING BLOCK.....	H-5
H.3.3.1 createTone.....	H-5
H.3.3.2 startTone.....	H-5
H.3.3.3 stopTone.....	H-6
H.3.3.4 stopAllTones.....	H-6
H.3.4 IO SIGNALS BUILDING BLOCK.....	H-6
H.3.4.1 signaRTS.....	H-6
H.3.5 EXAMPLES.....	H-6
H.3.5.1 I/O Configuration Building Block.....	H-7
H.3.5.2 AudibleAlertsAndAlarms.....	H-8
H.3.5.3 Packet Instantiation for Analog Audio I/O.....	H-9
H.3.5.4 I/O Control Building Block.....	H-10
<b>H.4 SERVICE PRIMITIVES. ....</b>	<b>H-10</b>
H.4.1 I/O CONTROL BUILDING BLOCK.....	H-10
H.4.1.1 Set/Get Parameters Service.....	H-10
H.4.1.2 Get TxActive Service.....	H-11
H.4.1.3 enableRTSCTS Service.....	H-11
H.4.1.4 setCTS Service.....	H-12
H.4.2 I/O CONFIGURATION BUILDING BLOCK.....	H-12
H.4.2.1 Set/Get Configuration Service.....	H-12
H.4.3 AUDIBLE ALERTS AND ALARMS BUILDING BLOCK.....	H-13
H.4.3.1 Create Tone Service.....	H-13
H.4.3.2 Start Tone Service.....	H-13
H.4.3.3 Stop Tone Service.....	H-14
H.4.3.4 Send Beep Service.....	H-14

H.4.3.5 Start Beep Service.....	H-15
H.4.4 I/O SIGNALS BUILDING BLOCK.....	H-15
H.4.4.1 signalRTS Service.....	H-15
<b>H.5 ALLOWABLE SEQUENCE OF SERVICE PRIMITIVES.....</b>	<b>H-17</b>
<b>H.6 UTILIZATION OF I/O BUILDING BLOCKS .....</b>	<b>H-18</b>
<b>H.7 PRECEDENCE OF SERVICE PRIMITIVES .....</b>	<b>H-19</b>
<b>H.8 SERVICE USER GUIDELINES.....</b>	<b>H-19</b>
<b>H.9 SERVICE PROVIDER-SPECIFIC INFORMATION.....</b>	<b>H-19</b>
<b>H.10 IDL .....</b>	<b>H-19</b>
<b>H.11 UML .....</b>	<b>H-19</b>

## List of Figures

Figure 1. Service Definition Overview.....	H-2
Figure 2. IO Configuration Building Block.....	H-4
Figure 3. IO Control Building Block .....	H-4
Figure 4. Audible Alerts and Alarms.....	H-5
Figure 5. ioSignals Building Block.....	H-6
Figure 6. Example Device Interface .....	H-7
Figure 7. Alerts and Alarms Example.....	H-8
Figure 8. Example Packet Instantiation .....	H-9
Figure 9. Control Interface.....	H-10
Figure 10. IO Configuration Building Block .....	H-19
Figure 11. Audible Alerts/Alarms Building Block .....	H-20
Figure 12. IO Signals Building Block.....	H-20
Figure 13. Control Building Block .....	H-21

## List of Tables

Table 1. Cross-Reference of Services and Primitives.....	H-3
--	-----

## H.1 INTRODUCTION.

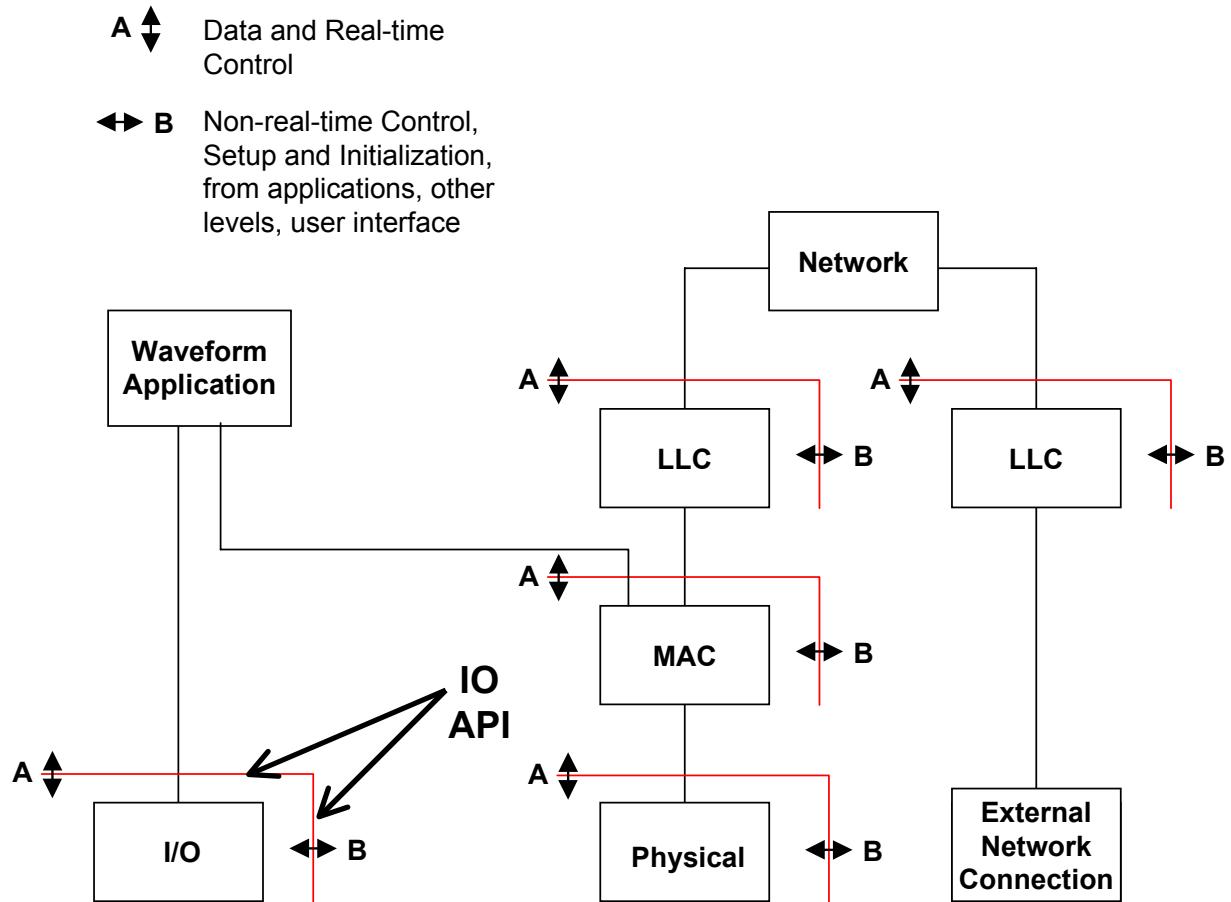
### H.1.1 Overview.

This document specifies SCA conformant building blocks for use by I/O service definitions. This document provides the definition of the building blocks utilized in the definition of an I/O Service provider. The I/O building blocks provide the following Services:

1. I/O Control: This building block provides the mechanism for the service user to control the operation of the I/O Device.
2. Audible Alerts and Alarms: This building block provides to the service user the ability to define and generate audible alerts and alarms to the operator.
3. I/O Configuration: This building block provides startup and runtime configuration of the device as necessary.
4. I/O Signals: This building block defines the signals that the I/O device will generate.

Definition of the data, and the data controls is defined by instantiating Packet Building block. When constructing a concrete API the Packet Building Block, along with the I/O building blocks are instantiated with the specific types utilized by the waveform(s).

The services defined in this building block are used in conjunction with other building blocks to generate the interfaces that form a complete application-programming interface (API). An example of generating interfaces from building blocks and APIs from interfaces is given in section 3 of this document.



**Figure 1. Service Definition Overview**

#### H.1.2 Service Layer Description.

The I/O Building Blocks define the fundamental structure for defining an interface for SW components to communicate to a specific type of I/O device. All I/O APIs shall use the I/O API building blocks. The I/O Building Blocks are defined as:

- I/O Control
- I/O Configuration
- AudibleAlertsAndAlarms
- I/O Signals
- Packet Building Block (not provided in this document)

#### H.1.3 Modes of Service.

There are two modes of service defined at this level for the I/O building block: Transmit mode and Receive mode. Further definition of service modes is left up to each implementation of an I/O API.

#### H.1.4 Service States.

The service states can not be defined until the I/O building block is instantiated for a specific API.

#### H.1.5 Referenced Documents.

JTRS-5000SCA, “Software Communications Architecture Specification (SCA)”,  
V2.2.1, April 30, 2004

### **H.2 UUID.**

Not applicable.

### **H.3 SERVICES.**

The features of the interface are defined in terms of the services provided by the Service Provider, and the individual primitives that may flow between the Service User and Service Provider.

The services are tabulated in Table 1 and described more fully in the remainder of this section.

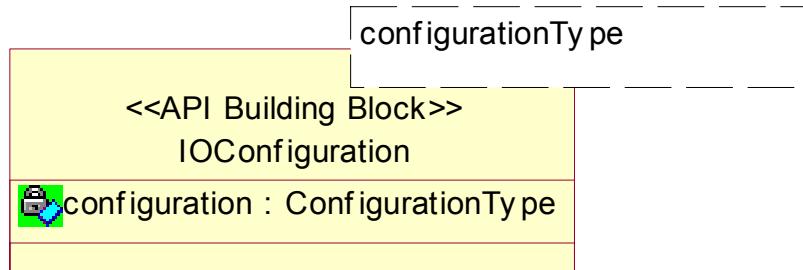
**Table 1. Cross-Reference of Services and Primitives**

<b>Service Group</b>	<b>Service</b>	<b>Primitives</b>
I/O Configuration	Configuration Parameters	setConfiguration
		getConfiguration
I/O Control	Control Parameters	setParameter
		getParameter
	Flow Control	getTxActive
		enableRTSCTS
		setCTS
I/O Signals	FlowControl	signalRTS
AudibleAlerts And Alarms	Tones	createTone
		startTone
		stopTone
		stopAllTones
	Beeps	createBeep
		sendBeep

#### H.3.1 I/O Configuration Building Block.

The I/O Configuration Building Block provides a generic interface for defining a specific I/O Device configuration interface. The interface details of each I/O Device are defined by providing a type definition to the ConfigurationType. This type may be a single element or a complex record structure. The auto generated “set” and “get” function for the resulting

“configuration” attribute is then used to configure the Device. Examples of configuration parameters are Sample Rate, BitRate, etc.



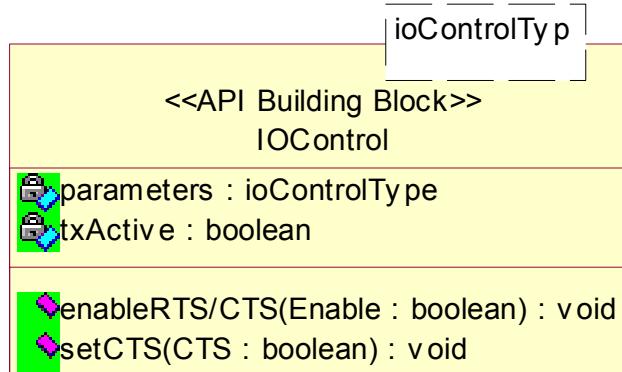
**Figure 2. I/O Configuration Building Block**

#### H.3.1.1 Configuration.

Configuration is used to configure the I/O device. Configuration is generic in order to provide a common interface to all I/O device types. This generalization requires that for any given type of device to be derived, an instantiation of the generic interface be provided, making a concrete interface. The configuration parameters are thus defined by the instantiation of the general I/OConfiguration building block.

#### H.3.2 I/O Control Building Block.

This Building Block defines the interface for controlling the operational parameters associated with a specific I/O Device. There is read and write operation associated with the “parameters” attribute, but only a read operation for txActive.



**Figure 3. I/O Control Building Block**

#### H.3.2.1 parameters.

This attribute defines parameters that are used to control the I/O device during operation. This attribute is read and write.

### H.3.2.2 txActive.

This attribute denotes whether or not the I/O device is actively transmitting data to the process that is downstream of the device.

### H.3.2.3 enableRTSCTS.

This method is used to enable and disable the RTS and CTS method of flow control. If enabled, the I/O device will invoke the RTS signal to make the downstream process aware that transmit data is forthcoming. The I/O device will not send data downstream until the setCTS method has been called indicating that CTS = True.

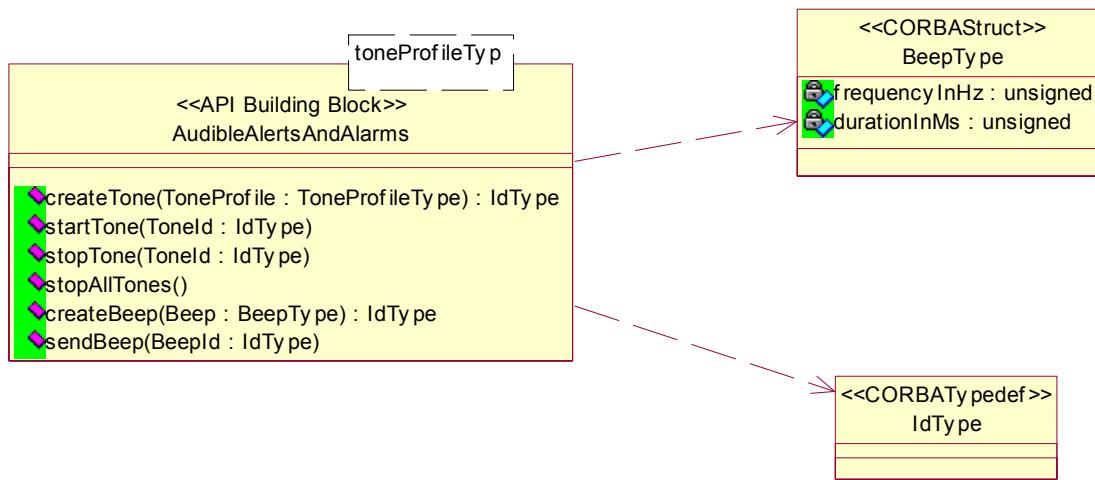
### H.3.2.4 setCTS.

This method is used by the I/O device service user to tell the I/O device that data should, or should not be sent downstream.

## H.3.3 AudibleAlertsAndAlarms Building Block.

The AudibleAlertsAndAlarms building block provides Tone and Beep services to the operator, and TxActive services to the I/O service user. The definition of a tone is user defined at instantiation time. This allows for tone type other than single tones (multi-frequency tones, sirens, etc.).

A beep is defined as a single frequency tone with an associated duration.



**Figure 4. Audible Alerts and Alarms**

### H.3.3.1 createTone.

CreateTone provides the service of creating a tone with the specified profile, for future use by the service user.

### H.3.3.2 startTone.

StartTone provides the service user the ability to start the generation of a previously created tone to the waveform operator.

### H.3.3.3 stopTone.

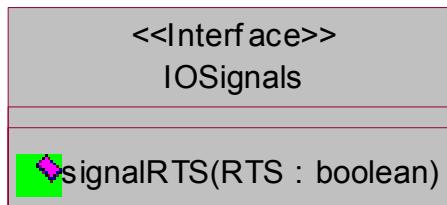
StopTone provides the service user the ability to stop generation of a previously started tone.

### H.3.3.4 stopAllTones.

StopAllTones provides the service user the ability to stop generation of all previously started tones.

## H.3.4 ioSignals Building Block.

This building block defines the signals that the I/O device will generate to the process connected downstream.



**Figure 5. ioSignals Building Block**

### H.3.4.1 signalRTS

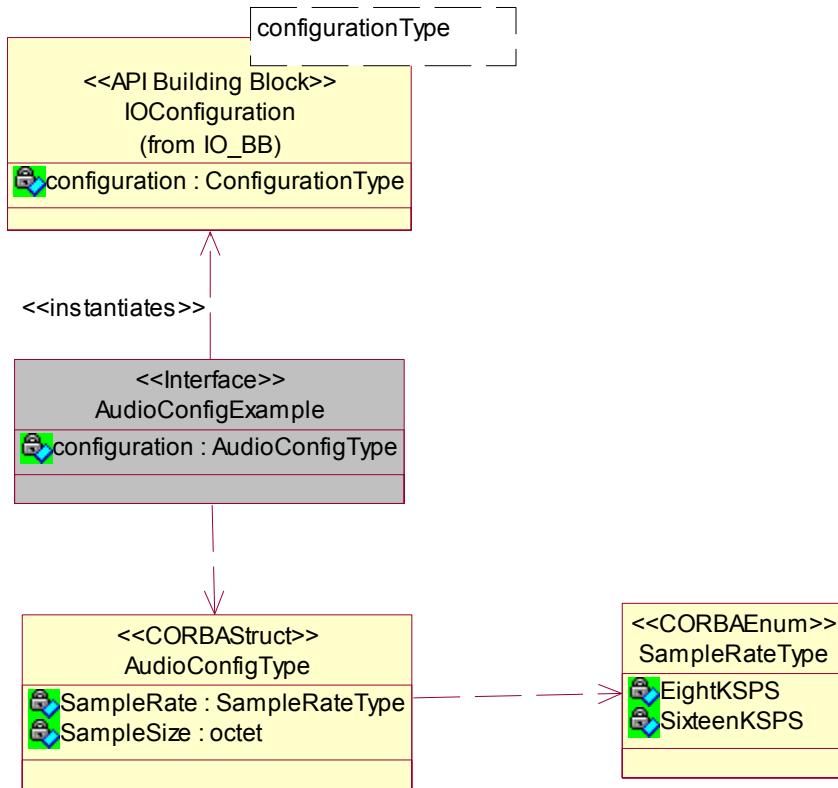
This method is used by the I/O device to signal the downstream component that the I/O device is ready to send data downstream.

## H.3.5 Examples.

The following examples are for demonstration of how parameterized and concrete building blocks are utilized in the generation of an I/O API for specific waveform requirements. Please note that UML Classes that are shaded gray are concrete interfaces useable for building an API.

### H.3.5.1 I/O Configuration Building Block.

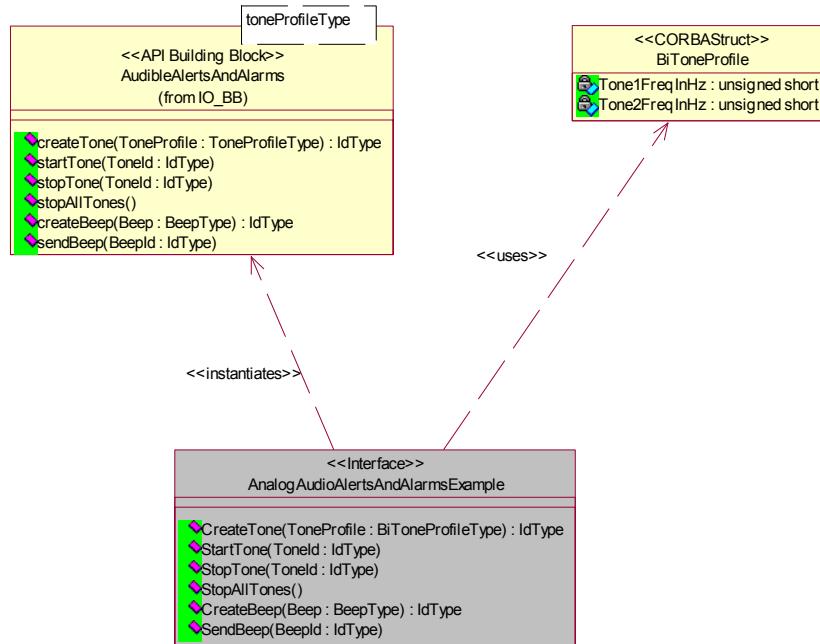
This building block is instantiated with parameters that are configurable. This interface is intended for non-real-time configuration that normally takes place at startup or during mode changes.



**Figure 6. Example Device Interface**

### H.3.5.2 AudibleAlertsAndAlarms.

This building block is instantiated into a usable interface by defining the ToneProfileType as a multi-tone record. This example interface may create single tones by setting only one frequency to a value other than 0. A more complicated structure could be used by instantiating the ToneProfileType as a sequence of single tones to implement a siren.



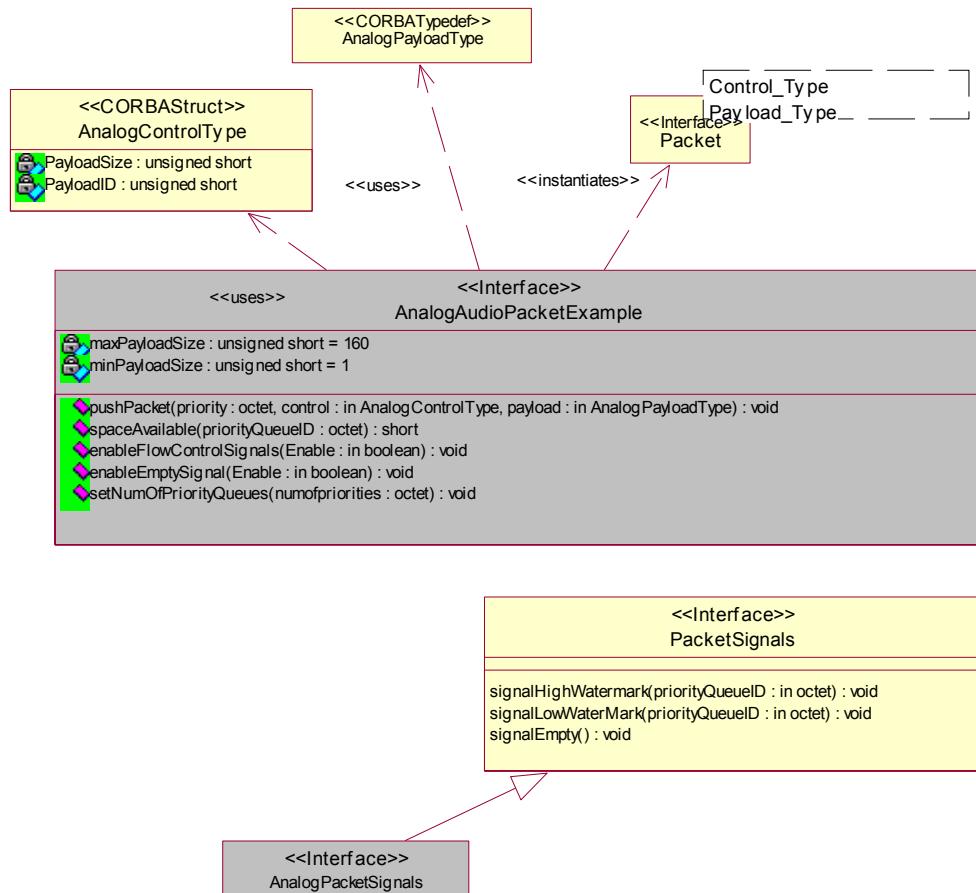
**Figure 7. Alerts and Alarms Example**

### H.3.5.3 Packet Instantiation for Analog Audio I/O.

This diagram shows the instantiation of the Packet building block and utilization of the Packet Signals building block. These building blocks are not defined in this document, but are utilized to define the data interface to/from a specific type of device.

The Signals interface is not instantiated, but inherited into a signal interface. This must be done because the Packet Signals building block is abstract, and must be inherited into a concrete interface.

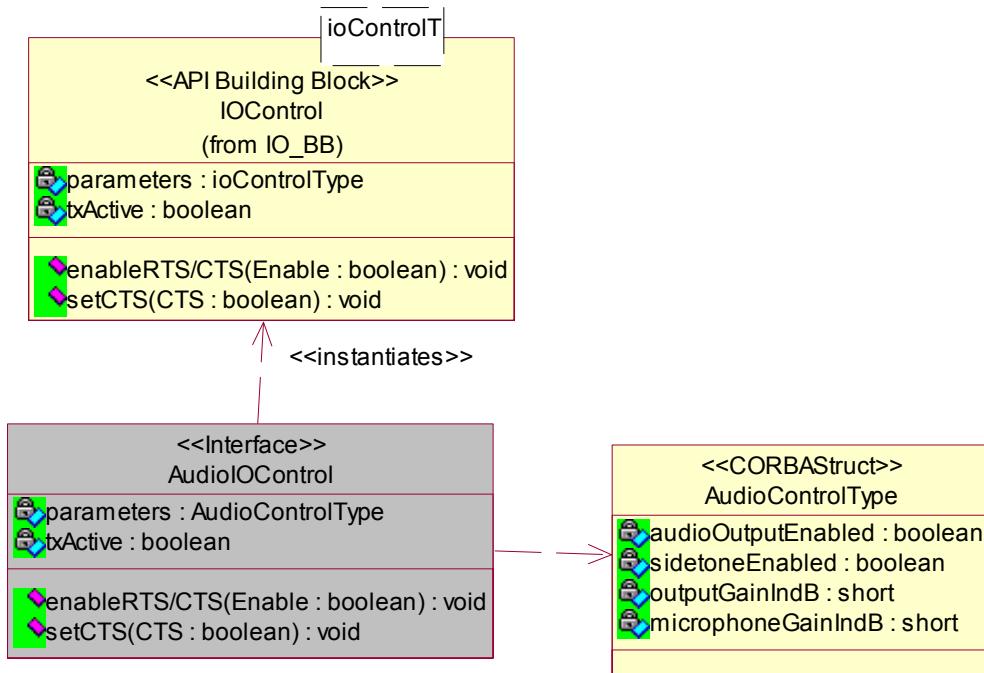
The Packet is instantiated with a specific Payload and Control type, suitable for implementing and analog I/O device.



**Figure 8. Example Packet Instantiation**

### H.3.5.4 I/O Control Building Block.

This building block is instantiated with those parameters that are used to control the I/O device during operation. The example given defines control suitable for an analog or digital Audio I/O device.



**Figure 9. Control Interface**

## H.4 SERVICE PRIMITIVES.

### H.4.1 I/O Control building block.

#### H.4.1.1 Set/Get Parameters Service.

##### H.4.1.1.1 Synopsis.

Parameters : ParametersType

“Parameters” is an attribute, whose “Get” and “Set” functions are auto-generated.

##### H.4.1.1.2 Parameters.

ParametersType is a template type that is user defined at instantiation time.

##### H.4.1.1.3 State.

Any valid state as defined by the “parameters” parameter.

##### H.4.1.1.4 NewState.

Any valid state as defined by the “parameters” parameter.

H.4.1.1.5 Response.

It is the service user's responsibility to verify the parameters were "set" via the "get" operation.

H.4.1.1.6 Originator.

Waveform Application.

H.4.1.1.7 Errors/Exceptions.

None defined.

H.4.1.2 Get TxActive Service.

H.4.1.2.1 Synopsis.

txActive : boolean

"txActive" is an attribute, whose "Get" function is auto-generated.

H.4.1.2.2 Parameters.

None.

H.4.1.2.3 State.

Any valid state.

H.4.1.2.4 NewState.

No state change.

H.4.1.2.5 Response.

The state of the devices transmit function is returned.

H.4.1.2.6 Originator.

Waveform Application.

H.4.1.2.7 Errors/Exceptions.

None defined.

H.4.1.3 enableRTSCTS Service.

H.4.1.3.1 Synopsis.

oneway void enableRTSCTS (in boolean enable)

H.4.1.3.2 Parameters.

The Enable parameter determines whether the CTS/RTS Flow Control protocol is to be activated.

H.4.1.3.3 State.

Any valid state.

H.4.1.3.4 NewState.

Flow Control is either active or inactive.

H.4.1.3.5 Response.

None.

H.4.1.3.6 Originator.

Waveform Application.

H.4.1.3.7 Errors/Exceptions.

None defined.

H.4.1.4 setCTS Service.

H.4.1.4.1 Synopsis.

oneway void setCTS (in boolean CTS)

H.4.1.4.2 Parameters.

CTS defines whether or not the I/O device is clear to send data to the downstream component.

H.4.1.4.3 State.

Any valid state.

H.4.1.4.4 NewState.

Transmitting, if CTS is true, or Not Transmitting if CTS is false.

H.4.1.4.5 Response.

None.

H.4.1.4.6 Originator.

Waveform Application.

H.4.1.4.7 Errors/Exceptions.

None defined.

## H.4.2 I/O Configuration Building Block.

H.4.2.1 Set/Get Configuration Service.

This attribute provides the configuration variables for the I/O device.

H.4.2.1.1 Synopsis.

Configuration : ConfigurationType

Configuration is an attribute, whose “Get” and “Set” functions are auto-generated.

ConfigurationType is a generic parameter, thus the definition of the configuration variables is application defined.

H.4.2.1.2 Parameters.

The auto generated Get/Set functions will return/accept a value of type configurationType.

H.4.2.1.3 State.

Any operational state.

H.4.2.1.4 NewState.

Any operational state as defined by the configuration parameter.

H.4.2.1.5 Response.

None.

H.4.2.1.6 Originator.

Waveform Application.

H.4.2.1.7 Errors/Exceptions.

None defined.

### H.4.3 Audible Alerts and Alarms Building Block.

H.4.3.1 Create Tone Service.

H.4.3.1.1 Synopsis.

`idType createTone (in ToneProfileType ToneProfile )`

H.4.3.1.2 Parameters.

ToneProfileType is a parameterized type. This type is generic to allow for a variety of tones to be specified. This is necessary because different applications generate different types of tones (i.e., – single tone, alternating tone, multiple tones, etc.).

H.4.3.1.3 State.

Any operational state.

H.4.3.1.4 NewState.

The device adds the newly created tone to the tone database.

H.4.3.1.5 Response.

Upon tone creation, the device will return a non-zero, unique Id.

H.4.3.1.6 Originator.

Waveform application.

H.4.3.1.7 Errors/Exceptions.

A value of zero is returned if the tone could not be created.

H.4.3.2 Start Tone Service.

H.4.3.2.1 Synopsis.

`oneway void startTone (in idType ToneId)`

H.4.3.2.2 Parameters.

ToneId identifies the tone that is to be generated.

H.4.3.2.3 State.

Any operational state.

H.4.3.2.4 NewState.

The device begins generation of the selected tone, if not already started.

H.4.3.2.5 Response.

None.

H.4.3.2.6 Originator.

Waveform application.

H.4.3.2.7 Errors/Exceptions.

None.

H.4.3.3 Stop Tone Service.

H.4.3.3.1 Synopsis.

oneway void stopTone (in idType ToneId)

H.4.3.3.2 Parameters.

ToneId identifies the tone that is to be stopped.

H.4.3.3.3 State.

Any operational state.

H.4.3.3.4 NewState.

The device halts generation of the selected tone, if not already stopped.

H.4.3.3.5 Response.

None.

H.4.3.3.6 Originator.

Waveform application.

H.4.3.3.7 Errors/Exceptions.

None.

H.4.3.4 Send Beep Service.

H.4.3.4.1 Synopsis.

oneway void sendBeep (in BeepType Beep)

H.4.3.4.2 Parameters.

BeepId : idType

H.4.3.4.3 State.

Any current operational state.

H.4.3.4.4 NewState.

The device outputs the selected beep frequency to the operator for the duration defined in the CreateBeep operation.

H.4.3.4.5 Response.

Upon beep creation, the device will return a non-zero, unique Id.

H.4.3.4.6 Originator.

Waveform application.

H.4.3.4.7 Errors/Exceptions.

A value of zero is returned if the beep could not be created.

H.4.3.5 Start Beep Service.

H.4.3.5.1 Synopsis.

oneway void startBeep (in BeepType Beep)

H.4.3.5.2 Parameters.

BeepId identifies the Beep that is to be generated.

H.4.3.5.3 State.

Any operational state.

H.4.3.5.4 NewState.

The device begins generation of the selected beep, if not already started.

H.4.3.5.5 Response.

None.

H.4.3.5.6 Originator.

Waveform application.

H.4.3.5.7 Errors/Exceptions.

None.

**H.4.4 I/O Signals Building Block**

This interface defines the signals that the I/O device can generate to the downstream component.

H.4.4.1 signalRTS Service.

This service is used when flow control is active and the I/O device is ready to begin sending data to the downstream component.

H.4.4.1.1 Synopsis.

oneway void signalRTS (in boolean RTS)

H.4.4.1.2 Parameters.

RTS denotes whether or not the I/O device is ready to send data.

H.4.4.1.3 State.

Any valid state.

H.4.4.1.4 NewState.

Waiting for a CTS from the downstream component.

H.4.4.1.5 Response.

None.

H.4.4.1.6 Originator.

I/O device.

H.4.4.1.7 Errors/Exceptions.

None defined.

## **H.5 ALLOWABLE SEQUENCE OF SERVICE PRIMITIVES.**

Attributes: Attributes may be set and fetched in any order. There is no order of precedence. The parameterized radio parameters should not be set unless all defined values have been initialized.

Tones and Beeps: Tones and beeps must be created before they can be started. Starting an already started tone will have no effect on the device.

Sending an already active beep will extend the remaining beep duration by one complete beep duration.

Stopping already stopped tones will have no effect.

## H.6 UTILIZATION OF I/O BUILDING BLOCKS.

Services	SINCGARS SC-PT	SINCGARS All Except SC-PT	WDW NB/WB	HQ 1/2	LOS	HF ALE	DAMA	DASA	VRC-99	Total Users
I/O Configuration	Yes	<b>Yes</b>	Yes	Yes	Yes	<b>Yes</b>	Yes	Yes	Yes	9
I/O Control	Yes	<b>Yes</b>	Yes	Yes	Yes	<b>Yes</b>	<b>Yes</b>	<b>Yes</b>	Yes	9
I/O Signals	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	9
Audible Alerts And Alarms	Yes	Yes	Yes	<b>Yes</b>	Yes	Yes	<b>Yes</b>	<b>Yes</b>	TBD	8?

## H.7 PRECEDENCE OF SERVICE PRIMITIVES.

Precedence of primitives is left to service definitions of full APIs.

## H.8 SERVICE USER GUIDELINES.

Service User guidelines are left to service definitions of full APIs.

## H.9 SERVICE PROVIDER-SPECIFIC INFORMATION.

This appendix shall identify the information to be documented for each service provider implementation.

## H.10 IDL.

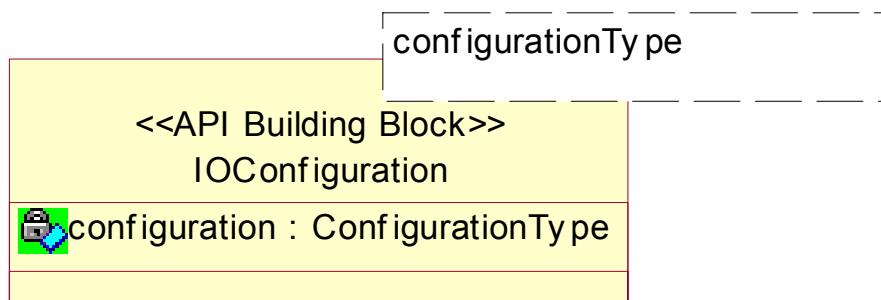
The IDL for an API is generated using the parameterized Classes of the building blocks to generate concrete classes with Waveform specific types and attributes. Parameterized Building Blocks documented herein is not intended to be instantiated directly in a UML diagram. The parameterized classes define the attributes and data types that are unique for each waveform.

The parameterized class is a template with which to generate user specific API definitions. To generate valid IDL from this Building Block, a concrete class must be generated that replaces the parameterized items with the user specific types and attributes. The completed UML diagram will not contain any reference to parameterized classes. Only concrete classes that were derived from them remain.

## H.11 UML.

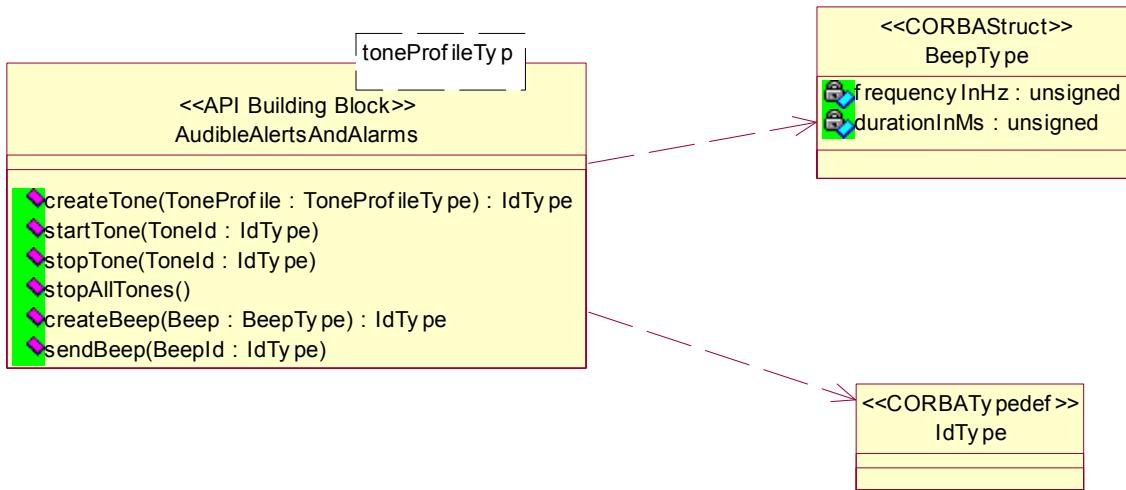
This appendix includes the class diagrams for the template and concrete building blocks utilized in generating a specific I/O API.

### 1. I/O Configuration Building Block.



**Figure 10. I/O Configuration Building Block**

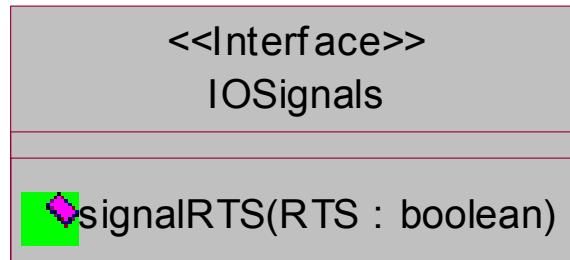
## 2. Audible Alerts and Alarms Building Block.



**Figure 11. Audible Alerts/Alarms Building Block**

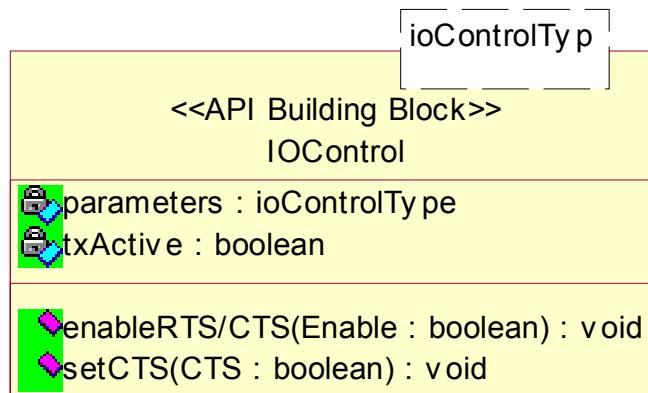
## 3. I/O Signals Building Block

This building block is not generic and does not need to be instantiated.



**Figure 12. I/O Signals Building Block**

#### 4. I/O Control Building Block



**Figure 13. Control Building Block**